Instituto Superior de Engenharia de Coimbra
Masters in Engineering and Industrial Management

Motor Overheating Monitoring Using Thermal Images And Artificial Neural Networks

Author

Raoni Rodrigues Pontes

Advisors

Prof. Doutor Mateus Mendes

Prof. Doutor Jorge Alexandre Almeida

Coimbra
April, 2020

RAONI RODRIGUES PONTES

Motor Overheating Monitoring Using Thermal Images And Artificial Neural Networks

Dissertation of course conclusion presented to the coordination of the Engineering and Industrial Management Course of ISEC as a requirement to obtain a masters degree in Engineering and Industrial Management.

Coimbra
April, 2020

I dedicate this work to everything I had to give up.

ABSTRACT

Motor overheating is a serious problem, which can be caused by overload, poor maintenance, age degradation, among other reasons. If necessary precautionary measures are not taken, it can result in premature damage of the motor or accidents. Monitoring the equipment condition, in this case the temperature, is a good way to prevent failures and enlarge its availability. The present work describes the outline of a system to monitor running motors using a thermal camera. The camera takes images of the most sensitive parts of the motor in real time. The images are then processed to extract the region where the temperature is higher. The region of the image at higher temperature is analysed. The area (as number of pixels) and shape of the region are then classified using an artificial neural network. The network is trained to recognize shapes where the safety of the motor could be compromised. All the algorithms to process images and artificial neural network development were written using Python tools. The model achieved an accuracy of 81.3% and 94.3% in the training and test sets, respectively.

Keywords: Motor Overheating, Thermal Image Processing, Artificial Neural Networks

Table of Contents

List of Figures

List of Tables

List of Symbols

h - Histogram

j - Synapse

k - Pixel gray level, Neuron

K - Pixel intensity

p - Normalized histogram

r - Range

s - Level

t - Time

x_j - Signal input

(x, y, z) - Spatial coordinates

w_{kj} - Synaptic weight

Ω - Frequency

ω - Correction coefficient

σ - Standard deviation

π - Phi

λ - Wavelength

Chapter 1

1. Introduction

In the industrial sphere, maintenance has always been a key point for productivity as well as costs. Maintenance carried out with planning guarantees the integrity of the equipment, the operators and the space occupied. Among the most applied techniques, the most efficient is predictive, as it monitors the equipment and maintenance is only done when it is really necessary. In certain situations, monitoring in person or with the installation of sensors is a complicated and sometimes unsafe task to be performed.

Aiming at continuous monitoring and the safety of the people involved, this work analyzed some non-destructive maintenance techniques, the condition to be chosen was the facility for detecting the overheating of internal combustion engines and could be monitored remotely through an artificial neural network. Thermography was the technique chosen because it has a relatively low cost compared to other options and allows work without the need to install sensors directly on the engine.

The present work was developed at ISEC facilities and using its equipment and laboratories. All algorithms that were developed for image processing and tools used for neural networks were implemented using free software. The model proposed was published in the International Symposium on Ambient Intelligence and Embedded Systems [1] that occurred on September 11 to 14 of 2019.

As it is a multidisciplinary work, the state of the art presents knowledge regarding image processing, maintenance application, thermography and artificial neural networks. After that, the methodology used is presented, showing the flow of the experiment, the equipment involved and the softwares used. The data capture experiment is described in detail, with a description of each step and the conditions found.

At the end, the results obtained are presented and discussed, also the description of the applied performance evaluation metrics. Promising results were achieved, which

were explained in the conclusion. In this last section, guidelines are presented for future improvements in a possible continuity work.

The remainder of the document is organized as follows. Chapter 2 contains all the basic knowledge that supports this work. It is possible to perceive the interdisciplinarity that surrounds the model, ranging from image processing to artificial neural networks. In Chapter 3 there is the applied methodology, showing the architecture of the experiment, the equipment and softwares used. Chapter 4 describes in detail the conditions, the experiment and the procedures performed. Chapter 5 shows all the development of the algorithm. Since the treatment of images to the artificial neural network. In chapter 6 the results are presented and discussed, in addition the main contributions of this work. Finally, in chapter 7 the conclusion and final considerations are presented.

Chapter 2

2. Industrial Vision applied for Maintenance

2.1 Digital Image Processing

An image can be defined as a two-dimensional function $f(x, y)$, where x and y are spatial coordinates and the amplitude of f at any pair of coordinates (x, y) is called image intensity at that point. The x, y of the function f must be discrete and have finite quantities for the image to be called digital. A digital image is composed of a finite number of elements and each element has a value and a particular location (Figure 1). Each element of the image is called a pixel [2]. Digital image processing refers to image processing using a digital computer.

Figure 1. Digital image and intensities of some pixels [2].

Vision is the most advanced of all human senses and therefore images play a very important role in human perception. However, unlike human vision which is limited to the visible band of the electromagnetic spectrum, computers and imaging devices cover almost all electromagnetic spectrum from gamma waves to radio waves. Thus imaging devices can process images generated from ultrasound, electronic microscopy and computer generated images, among other techniques. Digital image processing has a wide and varied field of applications. Image processing can be divided into three different groups — low, medium and high level.

Low level — This type of pixel-level processing involves primitive operations such as image preprocessing for noise reduction, contrast enhancement and image

smoothing. It is characterized by the fact that both the input and output objects are images.

Medium level — This processing involves more advanced tasks such as segmentation, splitting an image into regions or objects, and recognizing individual objects. The input object is an image but the output can consist of a set of attributes extracted from the images, such as outlines and borders.

High level — This group involves the interpretation of image content in an attempt to perform cognitive functions normally associated with human vision. This type of processing is usually associated with image analysis.

2.1.1 Origin and Evolution

The first applications of digital imaging appeared in the media in the early 1920s with the Bartlane system for intercontinental submarine cable transmission. However, the origins of digital image processing date back to 1964, when lunar surface images were processed to compensate for distortions introduced during their acquisition. In parallel with space applications, digital image processing techniques began to be used in the medical field. The invention of Computed Axial Tomography (CAT) in the early 1970s was one of the most important milestones in the application of image processing in medical diagnosis. From 1960 to the present, image processing has evolved significantly and is no longer an essentially academic and laboratory discipline. For this, the generalization of digital computers with consecutively improved features was crucial. Today, image processing has become an essential and economically viable technology in numerous practical applications such as: Medicine; Remote sensing; Astronomy; Industrial inspection; Defense; Biometry; Biology; Surveillance; Space exploration; Document Analysis and Criminal Investigation.

2.1.2 Image Acquisition

Images can be acquired using different sensors and techniques. With few exceptions, all imaging techniques involve mapping a 3D scene into a 2D space. The end result of most sensors used for imaging is a continuous waveform voltage whose amplitude and spatial behavior are related to the physical phenomenon being measured.

To create an image, it is necessary to convert the acquired continuous information into digital format. An image can be continuous with respect to *x* and *y* coordinates as well as amplitude. To convert it to digital format, it is necessary to have a sample of the function in the coordinates and in the amplitude. Thus, a digital image is obtained by a 2-D spatial sampling operation and intensity quantification of a continuous function. The number of pixels used to represent the image and the number of quantification levels used to represent the intensity of the pixels are the most important characteristics of an image.

2.1.3 Sampling

Spatial resolution is the definition of spatial places where image values are recorded. It is the discretization of the continuous domain, often defined in *x*, *y*, *z*, and *t*. A static image is a spatially varying two-dimensional signal. The sampling period, according to Nyquist's criterion, must be less than or at most half of the period of finest detail in an image [3]. If the image is over-sampled or exactly sampled it is possible to reconstruct the image without loss. If the image is undersampled, there will be spectral overlap resulting in a smoothing effect [3]. If a function *f* (t) does not contain frequencies greater than *w*, it is completely determined by giving its ordinates a series of points spaced by $1/(2\Omega)$ seconds. That is, if an image is sampled at a rate higher than the Nyquist frequency, an analog image can be retrieved from the sampled image with a zero-convergent error as the number of samples approaches infinity. The optimal resolution depends on the type of information we need to extract from the image. If technologically possible, it can be acquired and stored at the highest possible resolution. However, a reduction in resolution may be desirable not only to increase efficiency and reduce costs, but not to make analysis difficult. Figure 2 illustrates the effect of sampling at different sampling rates.

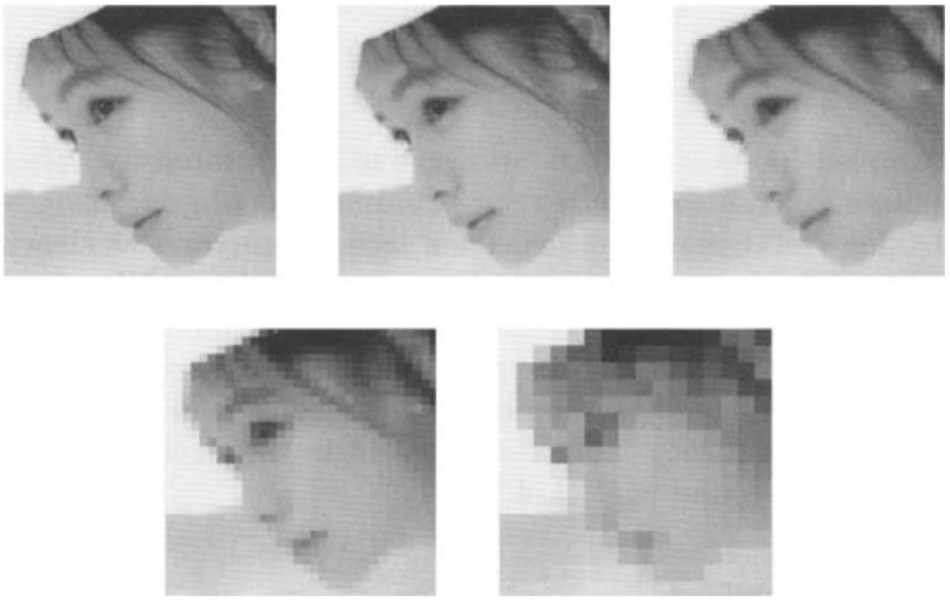

Figure 2. Images sampled at 256x256, 128x128, 64x64, 32x32, and 16x16 rectangular sampling grids [3].

2.1.4 Quantization

Discretization is the observation of a continuous signal in space or time at discrete positions or moments. Quantification is the assignment of discrete values to a range of continuous values. It is a complementary process to discretization. Quantization is the process of color discretization, allowing the conversion of an image with a continuous set of colors into an image with a discrete set of colors. This involves assigning a value to each sample so that the reconstructed image from the quantized samples is of good quality and the quantization-induced error is reduced. The dynamic range of values that image samples can assume is divided into a finite number of intervals and each interval is assigned to a single level [3]. Typically, the number of levels of each pixel in a gray image is represented by 8 bits, corresponding to 256 quantization levels, although some applications use 16, 24 or even 32 bits, where gray level range enhancement is required.

2.1.5 Imaging Operations

Image operations are used to perform various processing tasks such as: image enhancement; noise elimination; binarization; edge detection; geometric transforms; histogram calculation; projection calculation; transform calculation; etc. These operations can be local or global.

2.1.6 Histogram

The histogram is a representation of the frequency distribution and occurrence of a set of numbers. The histogram of a digital image with gray levels in the range [0, L-1] is a discrete function $h\ (rk) = nk$, where rk is the gray level ko and nk is the number of pixels in the image with that value of intensity. The histogram primarily provides the overall description of the image. If the image histogram is narrow, it means that the image is barely visible, because the difference in gray levels in the image is small. Even distribution of gray levels in a histogram means higher contrast and better visibility [3]. Normalization of a histogram is done by dividing each of its values by the total number of pixels in the image, n. This is given by $p\ (rk) = nk / n$, for $k = 0.1, ..., L-1$. This gives us an estimate of the probability of a gray level rk occurring. There are some indicators that can be taken from the histogram, such as overall intensity level, dynamic range, contrast, static information (mean, standard deviation, etc.) and other useful information for other image processing applications such as compression and segmentation. Figure 3 shows a histogram example acquired from the blue plan of a sample image.

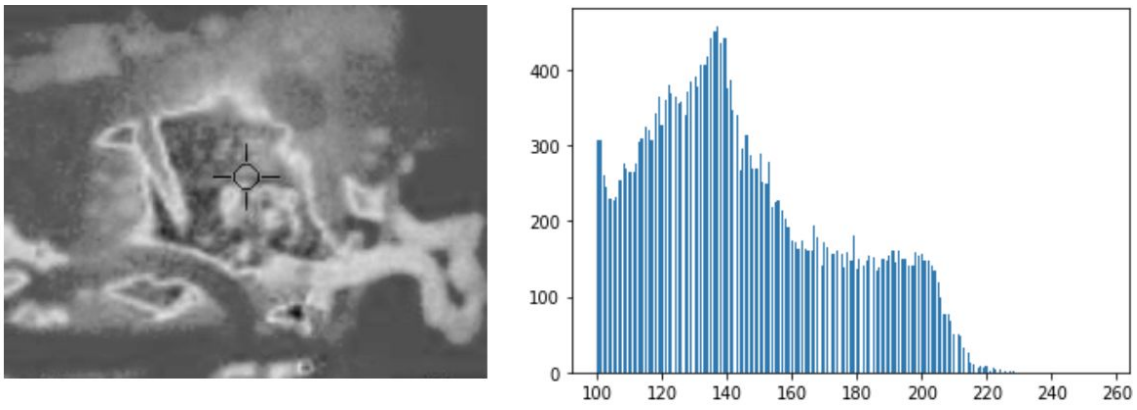

Figure 3. Histogram for blue scale picture, acquired from Python.

The basis of various spatial domain image processing techniques is the histogram. Histogram manipulation can be used for image enhancement. A low contrast image has a histogram that will be narrow. A high contrast image, the intensity values of each pixel, cover a wide range of values and the distribution of pixels is almost uniform.

2.1.7 Histogram Equalization

Histogram equalization is a technique of adjusting the grayscale of an image so that the gray level histogram of the input image is mapped to a uniform histogram [3]. Thus, the aim of histogram equalization is to obtain a uniform histogram from an initial image. r is a variable value in the range of [0, 1], which indicates the gray level of an image. When $r = 0$ refers to the black color while $r = 1$ refers to white, the transformation is shown below:

$$s = T(r) \qquad \text{(Eq. 1)}$$

This equation returns a level s for each pixel in the original image. It is assumed that $T(r)$ is a single value function, increasing monotonically in the range of [0, 1], and T (r) takes values between 0 and 1. The first condition preserves the order of black to white in grayscale, and the second condition ensures that the function is consistent with the defined range for pixel intensity values [3]. The equalization of an histogram is shown at Figure 4 below.

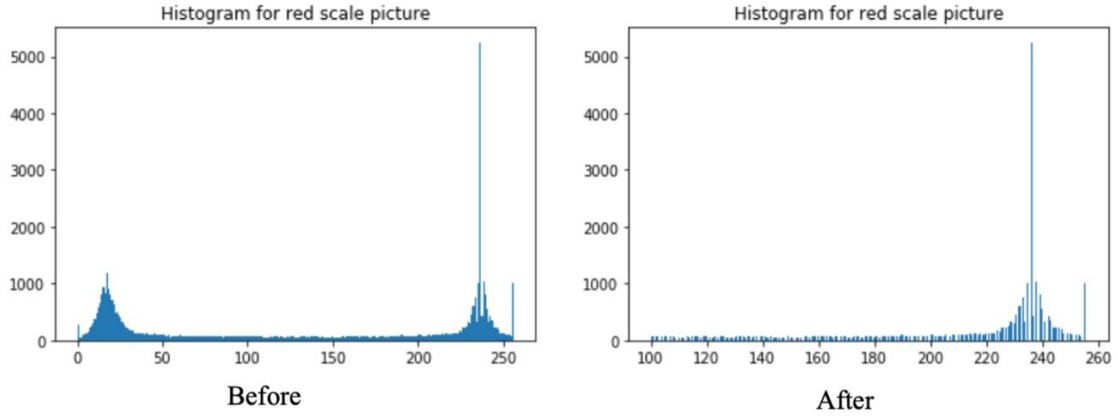

Figure 4. Histogram equalization.

2.1.8 Spatial Domain Filtering

Spatial domain filtering techniques are used for noise reduction in images [3]. Those techniques act directly on the pixels of an image. Consists of applying a filter mask at each point (x, y) of the image I, and the filter response at each point is calculated using predefined ratios. Thus, the application of these local operators consists of defining a central point (m, n), performing an operation that involves only the pixels in a predefined neighborhood around the central point, the result of which is the process

response and the value to be written in the pixel (*m*, *n*) of the filtered image. This process is repeated for each pixel of the image. In the case of spatial linear filtering the response is given by the sum of the products of the filter coefficients and the corresponding pixels calibrated by the filter mask. This process is similar to the concept of frequency domain filtering called convolution, and its filters are called convolution masks [2]. Convolution is called the process of calculating the intensity of a given pixel as a function of the intensity of its neighbors, and is based on weighting: that is, different weights are used for different neighboring pixels. The weight matrix is called the Mask, Convolution Window and sometimes Kernel. In general the linear filtering of an image *I* of size *M x N* with a mask of size *m x n* is given by the expression:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t)\, I(x+s, y+t) \qquad \text{(Eq. 2)}$$

where $a = (m-1) / 2$ and $b = (n-1) / 2$. Nonlinear spatial filters also operate based on the pixel neighborhood, and the way the mask traverses the image is the same as the linear filters. The filtering operation is conditionally based on pixel values in the vicinity of the considered pixel and does not use coefficients based on the sum of the products described above.

2.1.9 Gaussian Filter

A Gaussian filter is used to smooth the image which is applied to reduce noise [4]. The result of this operation is the blurring of the image, reminding a view through a translucent screen or as if viewed through an unfocused lens. Gaussian smoothing is widely used in the image preprocessing stage to enhance the image structure at different scales. Mathematically, the Gaussian filter is applied in the same way as the convolution of the image with a Gaussian function, such as a low pass filter, or as a simple average filter. This Gaussian function expresses the normal distribution in statistics. Since an image is defined in two dimensions, the Gaussian function used in the Gaussian filter applied to it must also be defined in two dimensions, namely one dimension in X and one dimension in Y. This is obtained in equation 3.

$$G(x,y) = G(x)\ .\ G(y)^t = \frac{1}{2\pi\sigma^2}\ e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad \text{(Eq. 3)}$$

This distribution has the format shown in Figure 4, where *x* is the distance from the origin on the horizontal axis and *y* is the distance from the origin on the vertical axis, and is the standard deviation of the Gaussian distribution of the pixels that make up the image.

The Gaussian filter used in image filtering is obtained by generating a Gaussian matrix that will be used as a mask in the convolution of the image. This matrix is generated using Equation 3, assigning the values of *x*, *y* and σ. Thus, for a matrix of dimension 5, with a standard deviation of 1 and both the *x* and y ranging from -2 to 2, the matrix structure consists of the values shown in Table 1.

Table 1. Gaussian filter example.

0.00078631	0.00655952	0.01330347	0.00655952	0.00078631
0.00655952	0.05472049	0.11097944	0.05472049	0.00655952
0.01330347	0.11097944	0.22507904	0.11097944	0.01330347
0.00655952	0.05472049	0.11097944	0.05472049	0.00655952
0.00078631	0.00655952	0.01330347	0.00655952	0.00078631

It can be seen from Table 1 that the highest value (the peak of the curve) is in the center of the table, and its neighbors are decreasing in value circularly, giving the idea of the bell shown in Figure 5 exactly as desired in Gaussian behavior. Depending on the application of this filter, it can still be normalized, i.e. have all its values divided by the average of all values in the matrix. Thus, it is not possible to obtain the values that make up this matrix, such as the standard deviation, from the matrix itself, that is, by supplying the parameter values of Equation 3 we get to Table 1, but the reverse is not true.

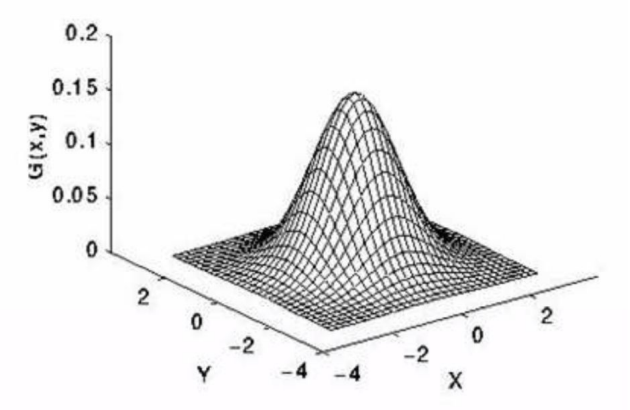

Figure 5. Bidimensional Gaussian curve [1].

2.2 Applications in Maintenance

When it comes to the use of infrared thermal imaging for automotive engines, there are only a few applications. In the area of Otto cycle motors, the use of infrared imaging has been used to detect variations in the temperature field in order to identify structural defects in the materials of casings, pipe lines, bearing housings and other engine parts. In that type of application, infrared image evaluation is usually performed manually by a specialist.

The branch of engineering that is concerned with methods of assessing and detecting faults in materials and equipment without damage is called Non-Destructive Testing (NDT). Knowing that equipment integrity can be affected by possible failures, NDT is a powerful tool that can be used in Predictive Maintenance ensuring safe operation, quality control and equipment life assessment. Failures may be cracks, leaks or variations in structural properties that may lead to loss of strength or service failure. NDT has no clearly defined boundaries. It can be performed using simple techniques, such as visual surface examination, and can also be done using geometric and dimensional analysis, radiography, noise comparison and thermography. There are still other methods that require direct intervention on the equipment, such as vibration analysis, mean effective pressure, ultrasonic and airflow at the intake and exhaust systems.

Barelli et. all use an NDT methodology based on the characterization of working conditions through acoustic and vibration measurements and relate the data to the

indicated mean effective pressure inside the cylinder [5]. Some indices are introduced in order to evaluate the engine working speed. The average indicated cycle, engine intake and exhaust valve lift level for different values of the produced electrical power are related to cylinder head vibration measurements and acoustic pressure levels in a close frequency range.

Sometimes some parameters are proposed in order to relate the data to the mean effective pressure indicated on the monitored cylinder, which indicates the operating state of the engine. Vibration measurement is a reliable methodology for assessing the quality of internal combustion engines, but requires the installation of sensors in the engine, which is not always feasible. Wavelet Packet Transform (WTP) is also a well-known way used for successful data and signal processing in condition monitoring and fault diagnosis. In this technique, the extraction of characteristics based on the transformation of wavelets is used, and the sound signals emitted by car engines are analyzed under fault and health conditions. The intention is to categorize the beeps into healthy and defective classes. Audible signals are generated from different car engines under healthy and faulty conditions. A problem with this technique is the noise generated by systems running parallel to the engine, which requires more arduous filtering work. Moreover, by the very nature of the signal generation at work, the detection of noise already represents a failure that is most likely linked to a considerable loss of efficiency [6].

The pressure signal on the cylinder can also be used to control the combustion engine cylinder filling process. The reason for this control is to optimize combustion in each cylinder in terms of performance, fuel consumption and emissions. The values obtained from the pressure signal are used to determine the descriptors directly involved in the control process, such as the indicated mean pressure, the maximum pressure or the crank angle to which half a fuel dose was burned. To calculate the values of these parameters with appropriate uncertainty, complex algorithms and thermodynamic models have to be developed along with the use of high processing power microcontrollers. Hence the need to look for easy fast methods of computing the combustion process descriptors: the beginning of the combustion angle or the angle corresponding to the maximum heat release rate. Some of the combustion parameters

can be evaluated based on the mean value and standard deviation of the measured pressure in the cylinder [7]. Those estimates are easy to calculate, but are useful only when parameter values follow a Gaussian distribution. The literature review shows that the sign of mean pressure deviations is also easy to determine and has a significant information load. Another interesting point is the exhaust gas analysis, which has two main focuses: an exhaust gas analysis and an acoustic signal in the exhaust flow. The contents of the exhaust cannot be used to estimate the quality of the combustion, nor problems with the injectors and engine lubricants [7].

Most exhaustion analysis is performed using a chemical analyzer score to record their component composition. In addition, exhaust flow monitoring with acoustics can also provide information on valve and injector problems, as well as other failures that affect combustion [8]. Although this technique has proven to be successful, it depends on intrusive exhaust flow measurement that is not feasible for industrial applications. The question of how viable it is in large engine applications due to the high exhaust temperatures that are in the region of 500 °C immediately after leaving is also present, providing potential high costs for sensors and their continuous replacement. Thermographic monitoring is a viable method, not only for its confirmed accuracy [9], but also for its ease of application for almost all desired monitoring conditions, whether static or dynamic situations.

2.3 Thermography

Capturing heat distribution images is called thermography. This kind of test is used from instantaneous fever detection to inspection of electrical and mechanical equipment. This inspection method is based on the fact that most components of a system show a temperature increase when they are defective. Temperature increase in certain regions of an engine may be due to loose connections or a worn bearing. By observing heat patterns in operating system components, faults can be located and their severity assessed. The inspection tool used by thermographers is the thermographic camera. These are modern devices that measure the natural infrared radiation emissions of a heated object and produce a thermal image, such as shown in Figure 6.

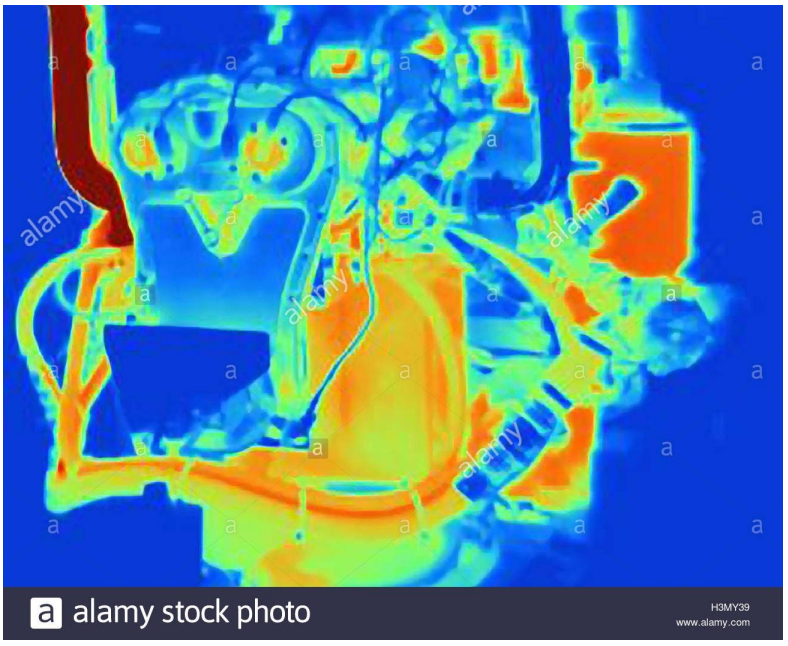

Figure 6. Thermal image. Credits at the photo.

Modern thermal imagers are portable with easy-to-operate controls. Because physical contact with the system is not required, inspections can be done under normal operating conditions, resulting in no production loss or downtime. An object when heated radiates electromagnetic energy. The amount of energy is related to the object's temperature. The Thermal Imager can determine the temperature of the object without physical contact by measuring the emitted energy.

2.3.1 Electromagnetic Spectrum

When an object is heated it radiates electromagnetic energy. The amount of energy is related to the temperature of the object. The thermal imager can determine the temperature of the object without physical contact by measuring the emitted energy [10].

A heated object radiates energy at different levels through the electromagnetic spectrum. In most industrial applications, it is the energy radiated at infrared wavelengths (λ) that is used to determine the temperature of the object. Figure 7 shows various forms of radiated energy in the electromagnetic spectrum, including X-rays, Ultra Violet, Infrared, and Radio. They are all emitted as a wave and travel at the speed of light. What differentiates the transmission modes is the wavelength that is related to the frequency [10].

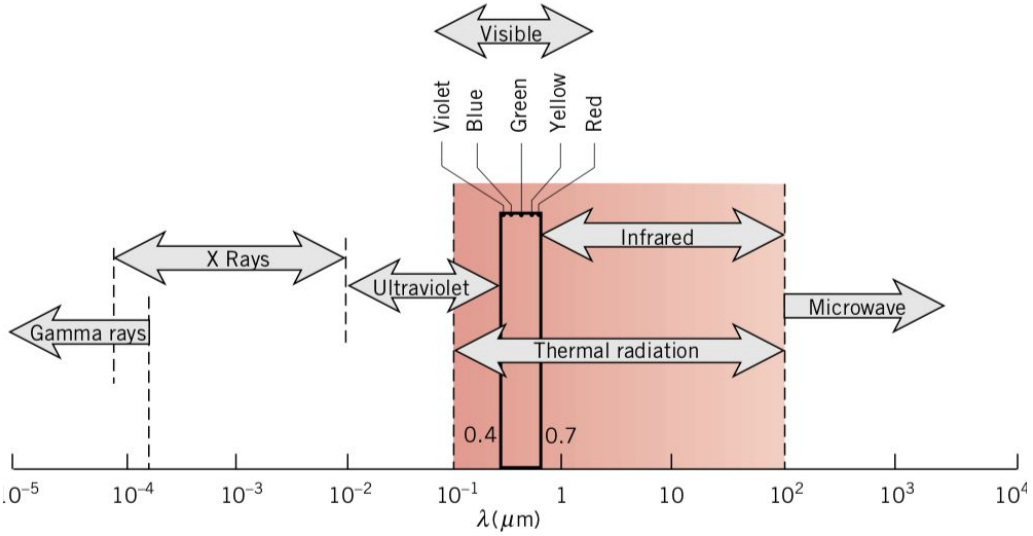

Figure 7. The Infrared region of the Electromagnetic spectrum [10].

Human vision can see visible light in the range 0.4 to 0.75 λ. Most infrared temperature measurements are made in the range of 0.2 to 20 λ. Since most emissions cannot be detected by a standard camera, the thermal imager can concentrate this energy through an optical system into a detector similar to visible light. The detector converts infrared energy into an electrical voltage that, after amplification and complex signal processing, is used to create the thermal image on the operator's display on board the thermal imager.

2.3.2 Energy Distribution

Figure 8 shows the energy emitted by a surface at different temperatures. As can be seen, the higher the target temperature, the higher the peak energy level. The wavelength at which the energy peak occurs becomes progressively smaller as the temperature increases. At low temperatures most of the energy is at long wavelengths [10].

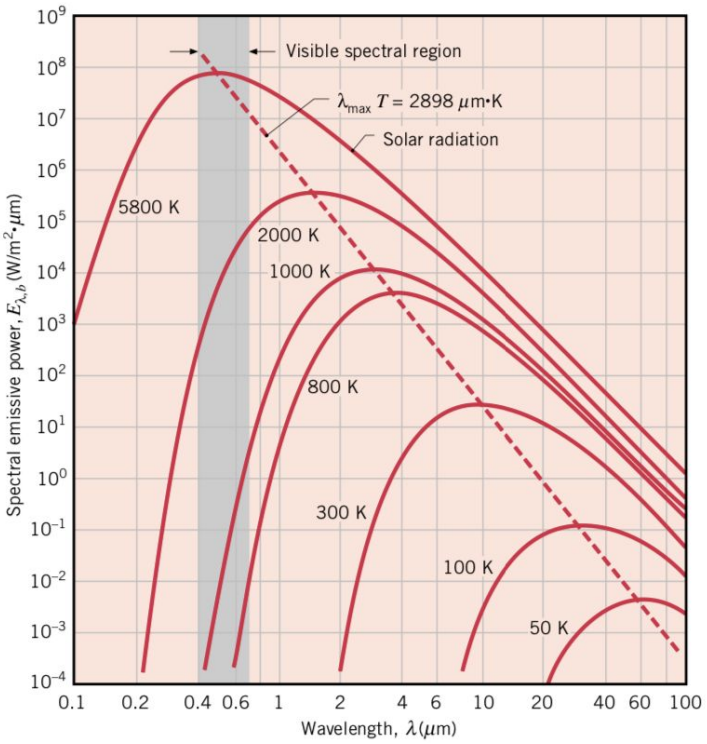

Figure 8. Infrared energy and distribution across the Electromagnetic spectrum [10].

2.3.3 Emissivity

Temperature and emissivity are the factors that determine the amount of energy radiated by an object. Blackbody is an object that has the ability to radiate the maximum energy possible for its temperature. In practice, there are no perfect emitters and surfaces tend to radiate a little less energy than a Blackbody [10].

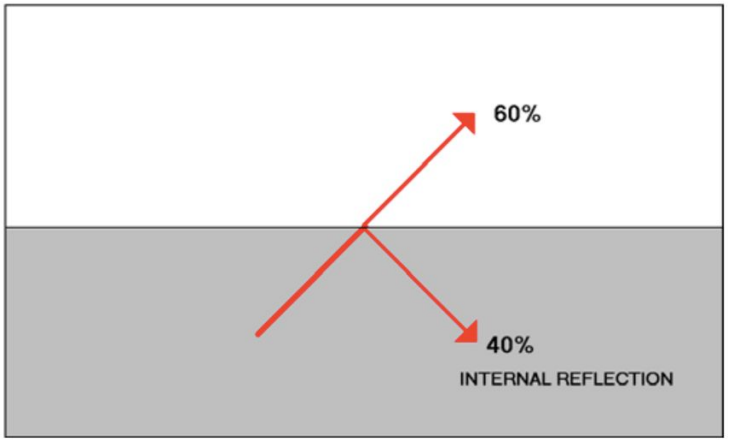

Figure 9. The Infrared energy reflected at a body surface [10].

Objects are not perfect emitters of infrared energy, as shown in the example of Figure 9. When energy moves toward the surface, a certain amount is absorbed and never escapes by reflection. From this example, it is clear that only 60% of the available

energy is actually used. Emissivity of an object is a ratio of the real quantity of energy radiated from the object and energy that it should emity if it were a Blackbody.

2.3.4 Thermography Images

Infrared - IR radiation may be felt by humans as heat, but it is not visible as shown in Figure 7. Thermal cameras, however, map the amount of IR radiation measured in a colour scale, in the range of colours visible to human beings. Among many cameras the scale of colours may vary or may be adjustable. In general, however, dark colours correspond to the lowest amount of radiation and temperature. The scale continues with blue, pink, red, orange, yellow and white.

Figure 6 shows an example of a thermal image of a running motor. As the image shows, different areas of the motor are at different temperatures. The blue areas are at lower temperatures. That means the surroundings of the motor are in general at low temperature. The exhaust pipe, at dark red, is at a temperature higher than the surroundings but not too hot. The hottest parts of the motor are clearly at the center bottom of the image, where the colours are already yellow.

The image of Figure 6 is stored in RGB format, with 8-bit depth. That means it is composed of three colour planes: one plane specifies the amount of red, the other the amount of green and the last the amount of blue for each pixel. And in each plane the amount of colour for each pixel is specified as an 8-bit integer. A black pixel will have zero in all planes. A white pixel will have 255 in all planes. The regions of higher temperature of the image will, therefore, have higher values for all planes.

2.4 Artificial Neural Networks

An Artificial Neural Network (ANN) consists of a connectionist structure in which processing is distributed over a large number of densely interconnected small units. This paradigm seeks to understand and emulate the properties arising from the high degree of parallelism and connectivity of biological systems. An ANN is composed of a large number of processor elements, the neurons, widely interconnected through links with a certain value that establish the degree of connectivity between them, called the weight of the connection or synapse. Thus, all processing is carried out

distributively between the processing elements of the network, where each one performs in isolation and in parallel, sending its result to other units through the connections between them. Although each neuron does quite simple processing, the association enables them to perform highly complex tasks [11].

The ability to solve a particular problem lies in the ANN architecture, that is, in the number and manner in which processing elements are interconnected, the weights of these connections, and the number of layers. To synthesize an ANN, there are some methods, the most commonly used of which is the empirical one, that generates an interconnected pattern to solve the problem through some training process capable of gradually modifying the weights to suit the particular problem. Synthesis is a determining factor in the success or failure of ANN training. Learning mechanisms make it possible to modify the interconnection pattern.

Learning in a neural network is closely related to the way humans learn their regular lives and activities. They perform an action and are accepted or corrected by a trainer or trainer to understand how to improve a given task. Likewise, a trainer is required so that neural networks can describe what should have been produced in response to input. An error value is calculated and sent back by the system based on the difference between the actual value and the value emitted by the network. The error value is analyzed and used to adjust the limit and weights for the next entry, for each layer of the network. As the network learns to analyze values, the error continues to become marginally smaller with each execution.

Also there is the test process, where network input patterns are presented, and the outputs obtained are compared to the desired outputs; and the application, where the synthesized network is used to solve a certain type of problem.

From a computational point of view, it can be said that in a neuron processing is performed on one or usually several inputs in order to generate an output. The artificial neuron is a logical-mathematical structure that seeks to simulate the shape, behavior and functions of a biological neuron. One can roughly associate the dendrite with the input, the sum with the processing and the axon with the output; therefore, the neuron is considered a fundamental information processing unit. Dendrites are the inputs, whose

connections with the artificial cell body are made through communication channels that are associated with a certain weight (simulating synapses). The stimuli captured by the inputs are processed by the sum function, and the triggering threshold of the biological neuron is replaced by the transfer function. Figure 10 schematically shows a McCullock and Pitts neuron [12].

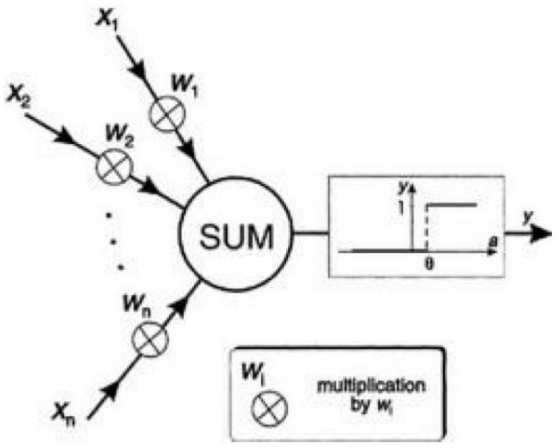

Figure 10. McCullock and Pitts Processing Unit Scheme.

The McCullock and Pitts neuron, while simple, is still used. In it the signals are presented to input x_j ; the input signal presented to synapse *j* connected to neuron *k* is multiplied by a synaptic weight w_{kj} ; In neuron *k*, the weighted sum of the received signals is produced, producing a certain level of activity [12]. If this activity level exceeds a certain threshold, the processing unit will produce a certain response as output, in this case *y*. Mathematically the model presented in Figure 10 can be represented by:

$$y = \sum_{j=1}^{n} w_{kj} . x_j \qquad \text{(Eq. 4)}$$

One of the earliest and simplest forms of neural networks was the perceptrons. In a perceptron, neurons are arranged in multiple layers, which can be classified into input layers, where patterns are presented to the network; hidden, hidden or intermediate layers, where most of the processing is done; and the output layer, where the completion of processing is presented. Usually it is possible to find the representation that will produce the correct mapping from input to output across

intermediate units. But it has been proven that at most two intermediate layers, with a sufficient number of units per layer, are required to produce any mappings that can be produced with a multi-layer perceptron. Neural Network inputs simulating stimulus uptake can be connected to many neurons, resulting in a series of outputs, where each neuron represents an output. These connections, compared to the biological system, represent the contact of dendrites with other neurons, thus forming the synapses [11].

The function of the connection itself is to turn the output signal from one neuron into an input signal from another, or to orient the output signal outwards. The different possibilities of connections between neuron layers can lead to different structures or architectures. Figure 11 represents one such architecture, in which an ANN with a hidden layer is presented.

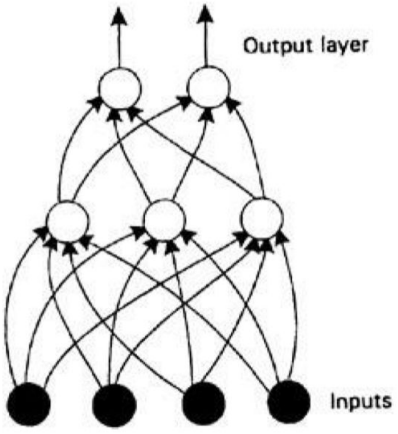

Figure 11. Example of a neural network.

One simple example of a neural network architecture is the Feedforward net. In this type of network the neurons are organized in layers. The input layer nodes communicate directly with the output layer or forward the information through the hidden layers (computational nodes).

The layered feedforward architecture has an organization similar to that of the human cortex, where neurons are arranged in parallel and consecutive layers, and axons always extend in the same direction. That means, information propagates from input to output, therefore there are no links between neurons of the same or previous layers. It is called a 1-layer network in reference to the output layer, because the input nodes process nothing, only present the defaults to the network [11].

2.5 Sensitivity Analysis

Binary classifiers are used to build artificial neural networks. These classifiers have some evaluation metrics, which help to better understand the results presented by neural networks. The most popular way to analyze a classifier is through the Confusion Matrix. It is a summary of the forecast results in a classification problem. Classification accuracy alone can be misleading if there is an unequal number of observations in each class or if there are more than two classes in the dataset. Because of that, it made a count of the correct and incorrect predictions and also divided by each class. This is the key to the confusion matrix. The confusion matrix shows how and where the classification model is confused when making predictions. It provides an insight not only into the mistakes made by a classifier, but, more importantly, the types of mistakes made. An example is shown at Table 2.

Table 2. Confusion Matrix

Predicted \ Actual	Class 1 = 0	Class 2 = 1
Class 1 Actual = 0	TP	FN
Class 2 Actual = 1	FP	TN

Where:

- Class 1 : Positive
- Class 2 : Negative
- Positive (P) : Observation is positive (for example: is a car).
- Negative (N) : Observation is not positive (for example: is not a car).
- True Positive (TP) : Observation is positive, and is predicted to be positive.
- False Negative (FN) : Observation is positive, but is predicted negative.
- True Negative (TN) : Observation is negative, and is predicted to be negative.
- False Positive (FP) : Observation is negative, but is predicted positive.

Going beyond the interpretation of the confusion matrix, comes another three important evaluation metrics: *Precision*, *Recall* and *F1-score*. In order to better understand each one, it is necessary to understand what each evaluation metric really means. Starting with *Precision*, the formula that determines is:

$$Precision = \frac{True\ positive}{Total\ predicted\ positive} \quad \text{(Eq. 5)}$$

Equation 5 shows how much precise/accurate the model is out of those predicted positives and how many of them are actual positives. *Precision* is a good measure to determine when the cost of false positive is high.

Moving forward, there is the Recall metric, which can be expressed through the equation:

$$Recall = \frac{True\ positive}{Total\ actual\ positive} \quad \text{(Eq. 6)}$$

Recall actually calculates how many of the actual positives the model captures through labeling it as positive (true positive). Applying the same understanding, *Recall* shall be the model metric used to select the best model when there is a high cost associated with false negative.

Finishing there is *F1-score*, which is a function of *Precision* and *Recall*. The formula that express this metric is the equation below:

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad \text{(Eq. 7)}$$

This means that when a balance between *Precision* and *Recall* is needed *F1-score* should be required. *F1-score* might be a better measure to use when there is an uneven class distribution (large number of Actual Negatives).

Chapter 3

3. Methodology

3.1 Experimental Setup

Figure 12 shows a diagram of the system. As the figure shows, the thermal camera is pointed to the motor. It is firmly positioned at a distance of the motor, so that it will not be affected by the motor vibrations or possible high temperatures that may be reached during operation of the motor. It is placed at a safe distance and pointing directly to the areas of the motor deemed more critical and, therefore, selected for monitoring.

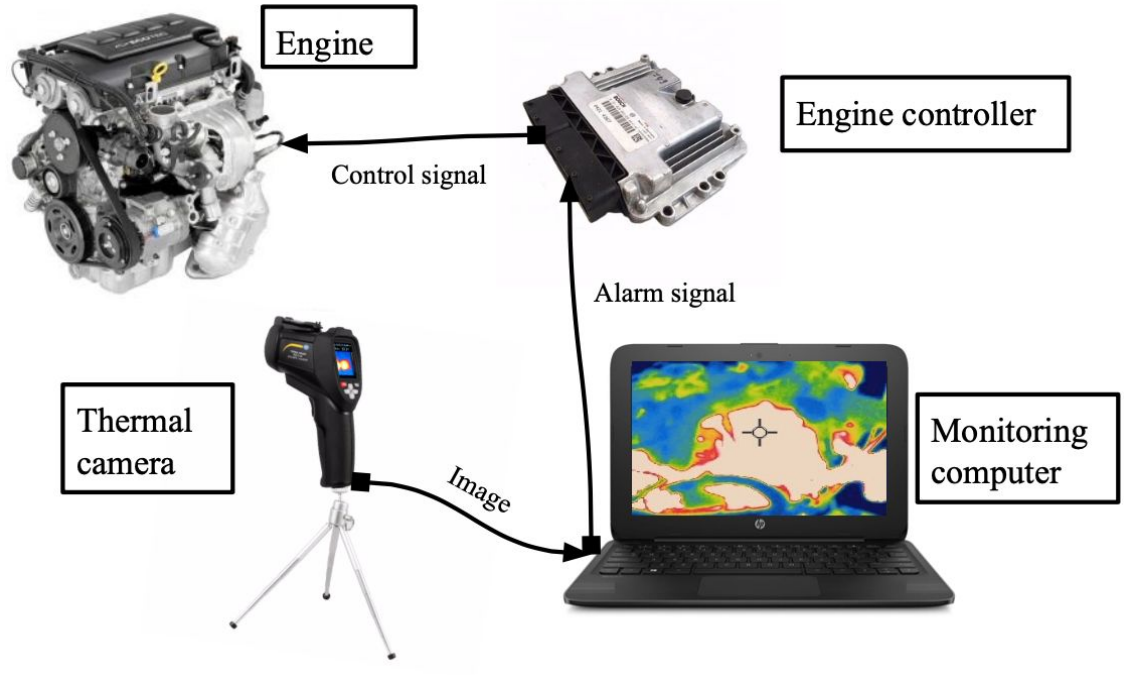

Figure 12. Block diagram of the system.

The camera captures pictures in real time and sends them to the computer via a USB link. The computer receives the images and processes them at the highest frame rate possible. When a number of consecutive images is detected as corresponding to a pattern where the motor is overheating, the computer sends a signal to the motor controller, either to reduce speed or to halt. The frame rate, the number of consecutive images that will raise an alarm and the actions to perform when an alarm is triggered will be defined by the engineer responsible for maintenance. The system was planned to

be as general as possible. So those parameters will be left as variables of the model to be configured for each possible deployment of the system.

3.2 Thermal Camera

The thermal camera used to run the test was a handheld FLIR™ E4, which has an uncooled microbolometer optical sensor that produces images with 240 × 240 pixels, standard JPEG format and is capable of capturing temperatures from -20 °C to 250 °C. Adjustment according to temperature is made automatically, and it has a frequency of 9 Hz. The link to the computer is made by a USB micro. The main reasons for choosing this camera were firstly the price and secondly the ease of handling to fix and point to the desired target. Figure 13 shows the camera used.

Figure 13. Thermal camera Flir™ E4 (Flir website).

3.3 Engine

ISEC labs have some engines available, two of which were considered for the experiments. The one used in the present experiments is fixed and connected to a dynamometer. It is a 1.4 L TU3JP engine, a four-cylinder gasoline engine developed by the PSA group (Peugeot Citroën) that has 74 hp and was developed in the early 90's. For a long time this engine model was applied into small and medium size vehicles assembled by the group. The fact that it has been used for a long time is considerable in the light of product durability and reliability. This engine is double overhead camshaft (DOHC or twin-cam) with 16 valves, which means it has two camshafts on the cylinder

head, one for the intake valves and the other for the exhaust valves as shown in Figure 14.

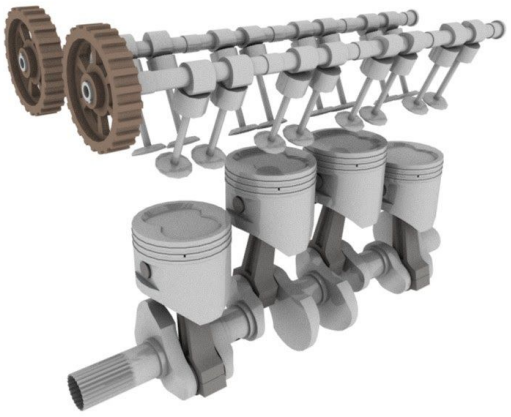

Figure 14. DOHC system.

Due to the large number of parts in constant friction and proximity to the combustion chamber, overheating in this region becomes a critical sign of a possible engine failure [13]. So that is why this region was the target during the tests.

3.4 Software Used

3.4.1 Ubuntu Operating System

Ubuntu is the most widely used desktop Linux operating system. It is based on Debian, another operating system created in the 1990s. Ubuntu development is led by British company Canonical Ltda, founded by Mark Shuttleworth in 2004. It is based on open source and its name comes from African languages. The name is often translated as "humanity to others" or "the belief in a universal sharing bond that connects all of humanity". The graphical interface is Unity as its default desktop environment for personal computers and smartphones. This interface is the heaviest of all other existing Linux interfaces, requiring about 1 GB of RAM and a modest processor. Ubuntu is a GNU / Linux operating system, so it is free and open source, it can be downloaded and installed for free accessing Ubuntu's official website http://www.ubuntu.com.

Ubuntu is by far the most secure operating system and the explanation is quite simple. Because systems using the Linux kernel are the least popular of all operating systems and have less faults to be explored. Due to this, crackers do not bother to create viruses for it. However, there are still anti-viruses to install. The Ubuntu operating

system is lighter compared to other operating systems such as Windows and Mac OS. It is so light that it can be run even from a USB stick, for example.

3.4.2 OpenCV

Before the explanation about this software, it is necessary to explain the concept of computer vision. Computer vision is machine vision, it is possible to obtain information from images, whether astronomical, microscopic or life size. It can be used with computational algorithms to describe and analyze the content of any digitized image. This practice is increasingly common in the industry in process quality control and robot orientation [14]. Computer vision is able to perform analyzes with precision and speeds that the human eye could not reach, bringing a new range of possibilities such as autonomous vehicle navigation, discovery of new planets and biological analysis in cells [15].

OpenCV (Open Source Computer Vision) is an open source programming library initially developed by Intel to make computer vision more accessible to developers and hobbyists. It currently has more than 500 functions, can be used in various programming languages (C++, Python, Ruby, Java…) and is used for various types of image and video analysis such as detection, tracking and face recognition, photo editing and videos, text detection and analysis [16].

3.4.3 Python

Python was created in 1990 by Guido Van Rossum at the Stichting Mathematics Center (CWI, see http://www.cwi.nl, last accessed 2020-03-27) in the Netherlands as a successor to the ABC language. Guido is remembered as the main author of Python but other programmers have helped with many contributions.

This platform is a high-level, object-oriented, interpreted programming language with dynamic semantics. The simplicity of Python reduces the maintenance of a program. Python supports modules and packages, which encourages modularized programming and code reuse. It is one of the fastest growing languages due to its compatibility (runs on most operating systems) and the ability to assist other languages.

3.4.4 Spyder

Spyder (formerly known as Pydee) is a powerful interactive development environment for the Python language with advanced editing, interactive testing, debugging, and introspection capabilities. Spyder can also be used as a library that provides powerful console related widget, and can be used to integrate a debug console directly into the layout of its graphical interface. Figure 15 shows what Spyder looks like.

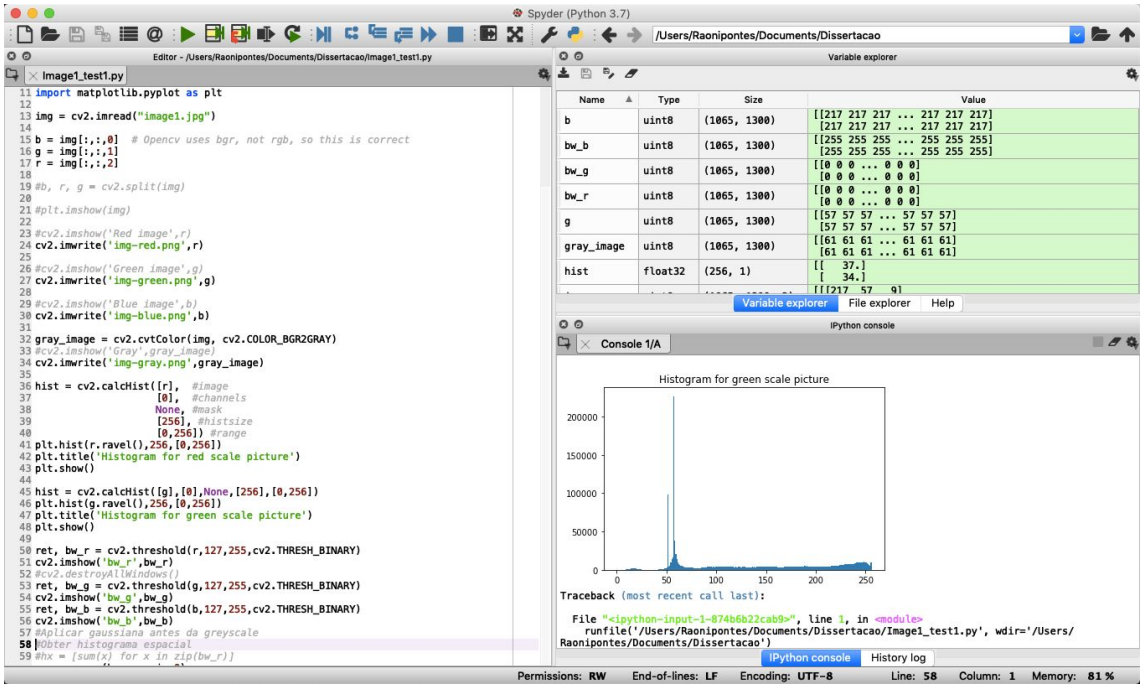

Figure 15. Spyder interface, acquired from Spyder.

3.4.5 Classification

Classification was performed using the Python Scikit-Learn library. Scikit-learn (originally scikits.learn) is an open source machine learning library for the Python programming language. It includes various sorting, regression, and grouping algorithms including support vector machines, random forests, gradient boosting, and is designed to interact with the NumPy and SciPy scientific and numeric Python libraries.

Chapter 4

4. Laboratory Experiments and Image Acquisition

As stated in item 3.3, the experiments were carried out within ISEC facilities in the internal combustion engine laboratory. Knowing the importance of the ambient temperature, this parameter was measured through two sources, the thermal camera itself and through the website https://www.accuweather.com. The test was performed on 2019-12-13. At this stage of the year, it was early winter and the room temperature was 13 ºC inside the laboratory. What can be said to be ideal, because at low temperatures, cooler areas captured in images make the hottest more evident. The camera was fixed at a distance of 1.5 m from the engine and set slightly above it. The focus was pointed towards the overhead cover, thus reducing the field of view and focusing only on the previously established part as the most critical. The experiment was designed to be performed in three phases and was done as follows:

1^{st} Phase — The engine was started and maintained at idle rotation speed (approximately 900 rpm) until it reached its constant working temperature. The engine started from 13 ºC and reached 59.5 ºC in approximately 10 minutes. During this time, images were taken at two minute intervals in order to perceive temperature growth in hot zones.

2^{nd} Phase — Rotation speed was rapidly raised to 2000 rpm and maintained until the engine reached constant working temperature. The engine started at 59.5 ºC and reached 76.8 ºC in approximately eight minutes. During this time, images were captured at 1.5 minute intervals in order to perceive temperature growth in hot zones.

3^{rd} Phase — As in the previous phase, the speed was rapidly increased to 4000 rpm and maintained until the engine reached its constant working temperature. The engine started from 76.8 ºC and reached 88.5 ºC in approximately eight minutes. During this time, images were captured at 1.5 minute intervals in order to perceive temperature growth in hot zones.

Throughout the test, even being connected to a dynamometer, the engine ran without load applied to the shaft. This is for safety reasons, as the engine is not enclosed by any safety. As it was perceived in the captured images, even without load, the thermal gradient between the parts of the images would be enough for a software recognition. Figures 16 (a), (b) and (c) respectively show an image of each phase 1, 2 and 3.

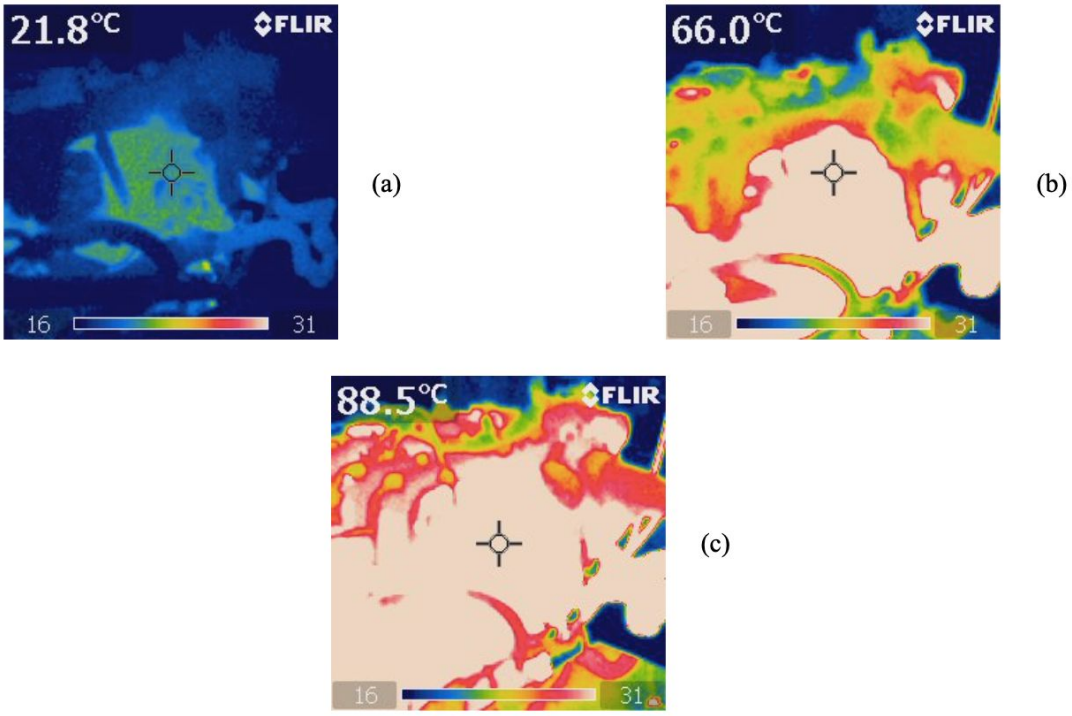

Figure 16. Images from each phase of the test.

At the end of the test, even with the difficulties of capturing images with a portable camera, 30 images were captured, 20 of which were related to good condition of use and 10 related to bad condition of use.

Chapter 5

5. Artificial Neural Networks Modeling

Starting the algorithm, the first thing to be done is to import the packages that will be used to assist the processing of the images. In this case, it started with the import of OpenCV called "cv2". Soon after, it is necessary to import NumPy, which is described in [17] as "fundamental package for scientific computing with Python"... "Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.". To facilitate the visualization of the obtained data and even help to compare results, the *matplotlib* package is imported. With these three firsts steps, the algorithm is ready to be started.

As the algorithm will work with a large set of images, the code must be contained within a function, which is executed several times at the same application. This function will be called "processa" and the code was named Alpha.

5.1 Dataset Augmentation

With the intention of applying an ANN to learn which condition of use is good or not for the engine, 30 images would be a relatively small package for this purpose. As a solution the data augmentation method was adopted. Data augmentation is a very powerful technique used to artificially create variations in existing images to expand an existing image data set. This creates new and different images from the existing image data set that represents a comprehensive set of possible images. Traditional transformations consist of using a combination of affine transformations to manipulate the training data [18].

For each input image, was generated a duplicate image that is flipped on the y-axis. Both image and duplicate are fed into the ANN. For a dataset of size N, a dataset of size $2N$ was generated. So now there are 60 images, 40 good examples and 20 bad examples. Figure 17 shows the result of this operation.

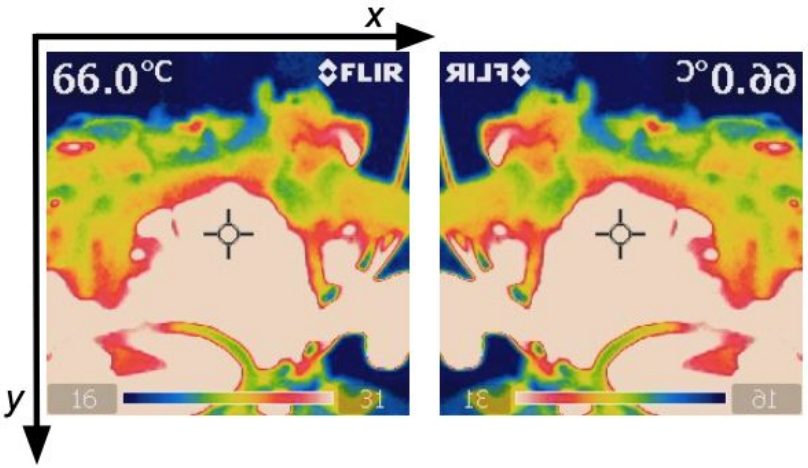

Figure 17. Image flipped on *y*-axis.

The code used for this operation was:

```
img = cv2.imread("1.jpg")
img2 = cv2.flip(img, 1)
cv2.imwrite('2.jpg',img2)
```

5.2 Image Processing

In order to avoid interference or counting error in the elaboration of the histograms for each plane, all images were cropped on the y-axis. This process removes from the top the indicated temperature and the camera brand, also from the base were removed the temperature range. Each image has 240 pixels at the *y*-axis, after the cut the algorithm will work only between the pixels 35 to 220. In this case *x*-axis remains integral with 240 pixels, the result is shown in Figure 18.

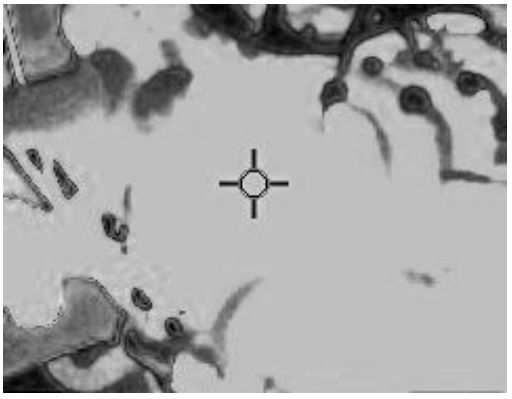

Figure 18. Blue plan cropped on *y*-axis, acquired from Python.

The next step was the separation of the plans and transformation to grayscale. This makes it easy to extract more specific histograms. For this, each cropped image had its plans separated through the order of interpretation of Python. Following the BGR sequence (blue, green and red) starting from zero to two respectively. Until now, the whole code is:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
def processa (Alpha):
    img = cv2.imread(Alpha)
    kernel = np.ones((3,3),np.float32)/9
    dst = cv2.filter2D(img,-1,kernel)
    crp = dst[35:220,0:240]
    b = crp[:,:,0]
    g = crp[:,:,1]
    r = crp[:,:,2]
```

A 3×3 Gaussian average filter were applied in order to smooth the images. For each pixel, a 3×3 window is centered on this pixel and all pixels falling within this window are summed up. The result is then divided by nine and applied. Due to the quality of the images generated by the camera used in the experiments, this process was effective and even helped with the convergence of the neural network.

5.3 Histogram Creation

To create a good database from the images, some histograms were extracted from the planes. Initially, histograms that represent the number of pixels for each intensity on a scale from 0 to 255 were extracted. This was done for each plane, so each

image contains three intensity histograms. A graphical representation can be seen in Figure 19.

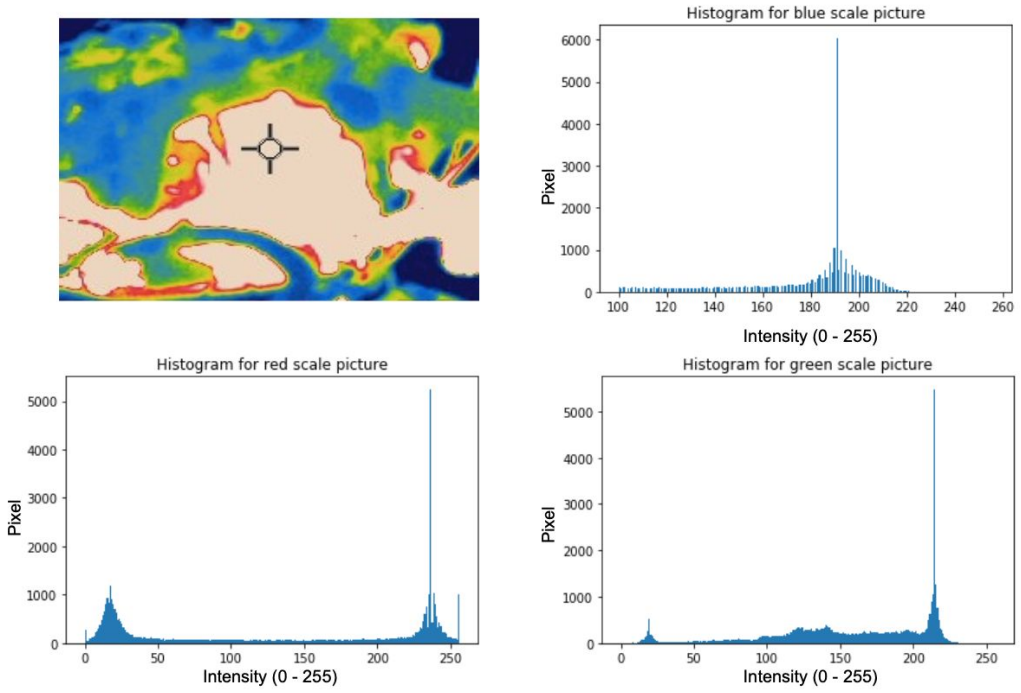

Figure 19. Intensity histograms (acquired from Python)

After that, vertical and horizontal histograms were extracted. For each axis, the program runs along the length and counts the pixel intensity and returns the total sum of them for each point on the axis. In the case of a 2D plane, the interaction of the vertical and horizontal histograms provides the exact position of each pixel, so it turns easy to identify the regions that contain the pixels with the greatest intensity, in this case those that are closer to white. Figure 20 shows the grayscale blue plane of an image captured during the tests and their respective sum histograms.

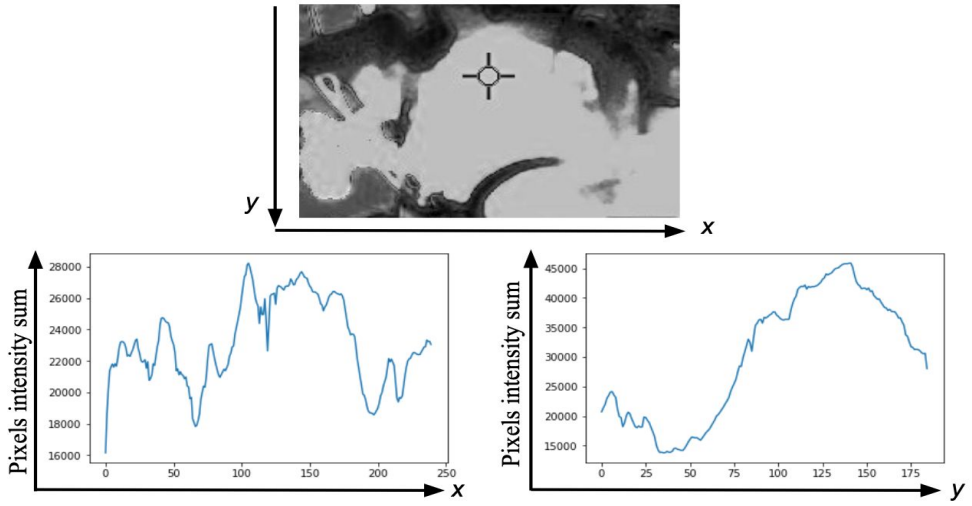

Figure 20. Horizontal and vertical histograms, acquired from Python.

The code used to make these two steps to acquire the histograms is described below:

```
hist = cv2.calcHist(([r],[g],[b]),[0],None,[256] e,[0,256])
hr = r.sum(axis=0) #X axle
vr = r.sum(axis=1) #Y axle
hg = g.sum(axis=0) #X axle
vg = g.sum(axis=1) #Y axle
hb = b.sum(axis=0) #X axle
vb = b.sum(axis=1) #Y axle
```

After this step, it was necessary to aggregate all histograms in a single variable, called total histogram (ht). In this way, each image will have an assigned vector containing all values referring to the intensities of its pixels and then the function "processa" could be closed. Below follows the code:

```
ht = hr
ht = np.append(ht,vr)
ht = np.append(ht,hg)
ht = np.append(ht,vg)
ht = np.append(ht,hb)
ht = np.append(ht,vb)
ht = np.append(ht,hist)
return ht
```

5.4 Features and Outputs

As previously mentioned, the images were separated into two groups of samples, one for good examples of operation and the other representing the overheating of the

engine. In order for the ANN to identify during the training which image represents good or bad functioning, a response value was assigned to each image, being zero for the good ones and one for the bads. For better understanding, a matrix [X,Y] was created containing 60 lines where in X there are the 1275 values corresponding to the (ht) of the images and in Y there are the answers assigned to each one of them. To make this happen the following code was used:

```
X = processa("1.jpg")
y = [0]
for j in range(2,41):
    Alpha = "%d.jpg" % (j)
    histo = processa(Alpha)
    X = np.vstack((X,histo))
    y = np.vstack((y,[0]))
for i in range(41,61):
    Alpha = "%d.jpg" % (i)
    histo = processa(Alpha)
    X = np.vstack((X,histo))
    y = np.vstack((y,[1]))
Y = np.ravel(y)
```

Due to the classifier restrictions that will be used, the (y) format had to be changed from a column to a row (Y) using the NumPy *ravel* function.

5.5 Artificial Neural Network Creation

As previously mentioned, a classifier provided by Python Scikit-Learn library was used. For this task, the MLPClassifier (Multi-layer Perceptron classifier) was chosen. Different from other classification algorithms, such as Support Vectors or Naive Bayes Classifier, MLPClassifier has an underlying Neural Network to perform the

classification task. Despite this, its implementation does not require more effort than the other classifiers.

Before starting the classifier, the database needs to be separated into two groups, one for training and the other for testing. Still using the same library, the tool *train_test_split* was used. This tool uses the default division of 75% for training and 25% of the dataset for testing. According to the results reported in [19], a fixed value of 20% appears to be a sensible choice for the test group, which means 80 percent for the training. Considering the relatively small size of the database, the standard division from *train_test_split* will be maintained, 75% for training and 25% for testing.

The MLPClassifier contains some parameters that need to be specified before it can work. The first one is the hidden_layer_sizes, here the i^{th} element represents the number of neurons in the i^{th} hidden layer. The second one is the maximum number of iterations. The solver iterates until convergence or this number of iterations. For stochastic solvers (as the chosen one ahead), this determines the number of epochs (how many times each data point will be used), not the number of gradient steps [20]. Going forward, alpha is the next parameter, which is a parameter for regularization term, also known as penalty term, that combats overfitting by constraining the size of the weights. The following step is to choose the solver, Stochastic Gradient Descent (SGD) was chosen because it is a simple yet very efficient approach for discriminative learning of linear classifiers under convex loss functions. Following is Verbose, a parameter used to print progress messages. Before finishing comes the random_state, which determines randomness of the process. For that case it remains off. And finishing comes the learn_rate, which is the learning rate schedule for weight updates. The code and values used for each parameter are shown below:

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
(trainX, testX, trainY, testY) = train_test_split(X,Y)
mlp = MLPClassifier(hidden_layer_sizes=(10,), max_iter=15, alpha=1e-4,
```

```
                solver='sgd', verbose=10, random_state=0,
                learning_rate_init=.1e-4)
```

With the model prepared it is possible to train it through the *fit* function. This step will fit the model to data matrix X and target(Y). It is possible to view the training and test scores to know the adjustment of the model using the *matplotlib.pyplot* tool called *score*. Below follows the respective code:

```
mlp.fit(trainX, trainY)
print("Training set score: %f" % mlp.score(trainX, trainY))
print("Test set score: %f" % mlp.score(testX, testY))
```

After completing all these steps, the algorithm is ready to apply the prediction. For this, predictions were made for four distinct groups, the first in the training dataset, the second in the test dataset, the third is a group, called *T* with ten random images from the dataset that were rotated in 2º in the counter-clockwise and the last one is the total dataset. Figure 21 below shows the group *T*.

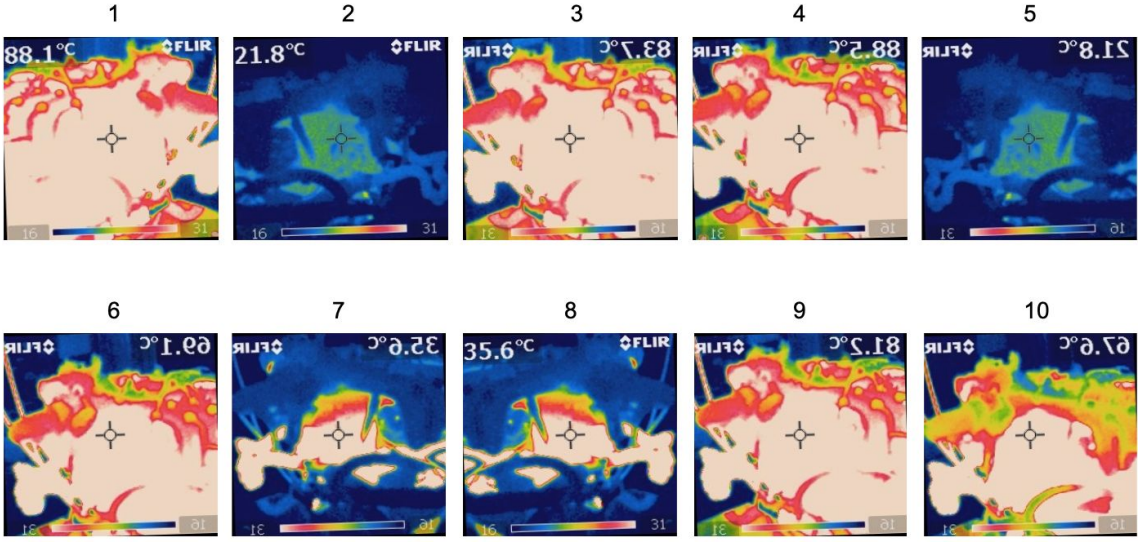

Figure 21. Group *T* with random images from dataset.

To assess the model performance, the results were extracted using the classification_report tool from the Python Scikit-Learn library.

```
pred = mlp.predict(T)
for m in range(len(T)):
      print("X=%s, Predicted=%s" % (T[m], pred[m]))
predt = mlp.predict(Tt)
for m in range(len(Tt)):
      print("X=%s, Predicted=%s" % (Tt[m], predt[m]))
predtrain = mlp.predict(trainX)
predtest = mlp.predict(testX)
#Training score
print(confusion_matrix(trainY,predtrain))
print(classification_report(trainY,predtrain))
#Test score
print(confusion_matrix(testY,predtest))
print(classification_report(testY,predtest))
#Global Test score
print(confusion_matrix(Y,predt))
print(classification_report(Y,predt))
```

As mentioned in section 2.5, it is important to know the performance metrics to better understand how the model is behaving.

Chapter 6

6. Results and Contributions

6.1 Results

After the entire neural network algorithm is completely ready, it was put to work. In the beginning, the output values did not converge and so the solution was to work with the ANN parameters, looking for a better adjustment. The values presented in the code at section 5.5 were those that presented the best results. The results presented by the algorithm is shown below:

```
Iteration 1, loss = 14.32722099
Iteration 2, loss = 0.52438455
Iteration 3, loss = 0.52463426
Iteration 4, loss = 0.52493428
Iteration 5, loss = 0.52526527
Iteration 6, loss = 0.52561253
Iteration 7, loss = 0.52596506
Iteration 8, loss = 0.52631473
Iteration 9, loss = 0.52665565
Iteration 10, loss = 0.52698372
Iteration 11, loss = 0.52729618
Iteration 12, loss = 0.52759131
Iteration 13, loss = 0.52786821
Training loss did not improve more than tol=0.000100 for 10
consecutive epochs. Stopping.
Training set score: 0.755556
Test set score: 0.933333
```

The report presented above shows that only 13 iterations were made, with no significant gain in the loss function and stopping the process. In addition, it obtained a mean accuracy score of 75.55% for the training group and 93.33% for the test group.

After that, the prediction for the *T* group in Figure 21 was made and the result is shown below. Table 3 shows the summary of the expected targets for each image in the group, the prediction made by the algorithm and accuracy reached.

```
X=[20581. 27080. 33922. ...0.0.0.], Predicted=1
X=[1623. 2517. 3066. ...   0.0.0.], Predicted=0
X=[11113. 14446. 17432. ...0.0.0.], Predicted=1
X=[15153. 21299. 25481. ...0.0.0.], Predicted=1
X=[1674. 2126. 2473. ...   0.0.0.], Predicted=0
X=[4839. 6218. 7476. ...   0.0.0.], Predicted=1
X=[1328. 1777. 2499. ...   0.0.0.], Predicted=0
X=[ 3546.  8137. 10270. ...0.0.0.], Predicted=0
X=[4945. 6064. 7222. ...   0.0.0.], Predicted=1
X=[ 7742.  8757. 10452. ...0.0.0.], Predicted=1
```

Table 3. Results from prediction on group *T*.

Figure	Expected Y	Prediction	Value
1	1	1	True
2	0	0	True
3	1	1	True
4	1	1	True
5	0	0	True
6	1	1	True
7	0	0	True
8	0	0	True
9	1	1	True
10	1	1	True
Total Accuracy			100%

Next, the prediction for the training dataset is performed, with the following result, first is the confusion matrix and then the classification report:

```
Training predict
[[ 6 11]
 [ 0 28]]
              precision    recall  f1-score   support

           0       1.00      0.35      0.52        17
           1       0.72      1.00      0.84        28

   micro avg       0.76      0.76      0.76        45
   macro avg       0.86      0.68      0.68        45
weighted avg       0.82      0.76      0.72        45
Average Total    0.813     0.733      0.72        45
```

For the training dataset, the confusion matrix is showing that 6 samples had the true positive, 28 samples had true negative against 11 with false positive and 0 with false negative.

The results from the prediction for test dataset is shown below:

```
Test predict
[[ 2  1]
 [ 0 12]]
                precision    recall  f1-score   support

           0       1.00      0.67      0.80         3
           1       0.92      1.00      0.96        12

   micro avg       0.93      0.93      0.93        15
   macro avg       0.96      0.83      0.90        15
weighted avg       0.94      0.93      0.93        15
```

```
   Average Total       0.943     0.896     0.92        15
```

For the test dataset, the confusion matrix is showing that 2 samples had the true positive, 12 samples had true negative against 1 with false positive and 0 with false negative.

Finally, the global test prediction is shown below:

```
Global predict
[[ 8 12]
 [ 0 40]]
                  precision    recall  f1-score   support
              0       1.00      0.40      0.57        20
              1       0.77      1.00      0.87        40

       micro avg       0.80      0.80      0.80        60
       macro avg       0.88      0.70      0.72        60
    weighted avg       0.85      0.80      0.77        60
   Average Total       0.843     0.766     0.763       60
```

For the global dataset, the confusion matrix is showing that 8 samples had true positive, 40 samples had true negative against 12 with false positive and 0 with false negative.

Considering the above predictions, the following results from the training group for the metrics *Precision, Recall* and *F1-score* are respectively 0.813, 0.733 and 0.72. Now for the test group, the predictions metrics are respectively 0.943, 0.896 and 0.92. Finally for the global dataset, the predictions are respectively 0.843, 0.766 and 0.763. Ideally, the perfect model will have the value of 1 for all these metrics, but that is next to impossible in real-world scenarios.

For a better visualization of the results, Table 4 shows a summary of the values obtained for each metrics in each group.

Table 4. Prediction results.

Dataset \ Metrics	*Precision*	*Recall*	*F1-score*
Training dataset	81.3%	73.3%	72%
Test dataset	94.3%	89.6%	92%
Global dataset	84.3%	76.6%	76.3%

Analyzing the results of the confusion matrices, it is clear that the model's algorithm never presents false positives, but always false negatives. Searching for the reason for this phenomenon, it becomes easier when the results obtained in the global dataset are related to the expected response for each image. It was observed that false negatives have always occurred in images related to the proper functioning of the engine, more specifically during the warm-up phase.

This can be explained by the way the thermal camera generates the images and how the colors are distributed and scaled. Despite having a good range of thermal capture, its high thermal capture setting prioritizes variations closer to the ambient temperature. In this way, higher temperatures are represented by colors closer to white, which has pixels with high values in the three planes. As the algorithm is based on learning pixel histograms, this lack of calibration in the variation of colors according to the temperature has strong indications of being the reason for such errors.

6.2 Main Contributions

The proposed model proves to be important due to the fact that it is a viable alternative for remote monitoring of engines or equipment subject to heating. In the face of a market where efficiency leads to high productivity and low production cost, this type of continuous monitoring aimed by artificial neural networks has a great potential to be well disseminated in the market. During participation in the Symposium AmiEs

[1], the model presented aroused a lot of interest from the participants. This reinforces the acceptance of the idea as a solution to a range of problems.

Obtaining an average accuracy of 86.6%, the model is effective in detecting overheating. The errors in the prediction are directly linked to the equipment involved. With that it can be concluded that the model works and is totally applicable to its initial purpose.

Specifically for this work, the model developed was applied in the detection of overheating of internal combustion engines, due to the wide application of this type of power generator. But since the algorithm is based on learning related to pixel histograms, it is easy to imagine an application expansion. In human activity there are countless other equipment subject to heating that need continuous monitoring, such as electrical and gas systems.

This work contributes to the equipment maintenance scenario. It brings up a new form of monitoring, based on the recognition of thermal images and classifying the functioning status of a system through an artificial neural network.

This model stands out from the other existing methods. It is not necessary to install sensors directly on the engines, it allows inexpensive continuous monitoring, allows ANN monitoring and can be performed remotely.

Chapter 7

7. Conclusion

Despite technological developments in the field of power generation, internal combustion engines are still the most widely used alternative. This work explained the need to create new predictive maintenance methods due to factors such as cost and safety. The alternative of using thermal images associated with neural network processing proves to be a viable and relatively low cost option when compared to current methods, in addition to being safer because it can be done remotely.

In this dissertation, a neural network model using Python Scikit-Learn library was explained and built. A dataset based on thermographic images of an Otto cycle engine was used and a classifier algorithm was created to predict the detection of overheating.

This model is achieving reasonable accuracy of 81.3% and 94.3% in the training and test dataset, respectively. It is observed that the accuracy of the model is superior to 80%. For the random group *T,* accuracy achieved 100%, which can be explained by the fact that the samples chosen have very distinct characteristics.

It was observed that the thermal camera used has colors close to white for warmer areas. Knowing that white has has pixels with high values for the three color planes, it can be concluded that this may have affected the accuracy of the model. This may be an indication of why the greatest number of errors occurred in images related to proper function but with high temperature zones. Improving accuracy can also be done through better images captured with better resolution equipment. The model can be further improved through cross-validation, resource engineering or changing the arguments in the neural network estimator.

The method proposed in this work showed itself with testing and validation that can be easily adapted to the industrial world, with improvements in the equipment involved and fine tuning of the algorithm, this system can be reliable enough to control engines in power generation plants, trains, ships and any application where prolonged uses are needed.

REFERENCES

[1] Raoni Pontes, Mateus Mendes, José Torres Farinha, Jorge Almeida, "**Motor Overheating Monitoring Using Thermal Images And Artificial Neural Networks**," 18th International Symposium on Ambient Intelligence and Embedded Systems, Coimbra Engineering Academy, Coimbra, Portugal, 11-14 September 2019.

[2] Gonzalez RC, Woods RE. Introduction. *Digital Image Processing*. 2 ed. New Jersey: Prentice; 2002; ISBN 0-13-168728-x 978-0-13-168728-8

[3] Acharya T, Ray AK. Image formation and representation. *Image Processing - Principles and Applications*. New Jersey: John Wiley & Sons, Inc.; 2005; ISBN-13 978-0-471-71998-4

[4] http://www.librow.com/articles/article-9, "Gaussian filter, or Gaussian Blur" , accessed on 03/01/2020.

[5] L. Barelli, G. Bidini, C. Buratti, R. Mariani. Diagnosis of internal combustion engine through vibration and acoustic pressure non intrusive measurements. Applied Thermal Engineering, Elsevier, 2010, 29 (8-9), pp. 1707, DOI: 10.1016/j.applthermaleng.2008.07.025.

[6] H. Ghaderi, P.Kabiri. Automobile engine condition monitoring using sound emission. Turk J Elec Comp Sci, 2017, 25 (1807-1826), DOI: 10.3906/elk-1605-77.

[7] Bakowski A., Radziszewski L., Świetlik P., Zmindak M., Analysis of information content of in-cylinder pressure signal deviations from the mean values, Technical transactions, Vol. 12/2018, pp. 151-158, DOI: 10.4467/2353737XCT.

[8] J. Jiang, F. Gu, R. Gennish, D. J. Moore, G. Harris, and A.D. Ball. Monitoring of diesel engine combustion based on the acoustic source characterisation of the exhaust system. Mechanical Systems and Signal Processing, 2007, Vol. 22 (6) / 2008, pp. 1465-1480, DOI: 10.1016/j.ymssp.2007.12.003.

[9] J. Monieta, The use of thermography in the diagnosis of ship piston in- ternal combustion engines, Poland, 2018, DOI: 10.1051/201818201027.

[10] Incropera F., Fundamentals of heat and mass transfer, 2007; ISBN-13 - 978-0-471-45728-2

[11] Gurney k., An introduction to neural networks, Taylor & Francis e-Library 2004. ISBN: 0-203-45151

[12] McCulloch w., Pitts w., A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biophysics 7, p.115-133.

[13] Heywood, John B., Internal combustion engine fundamentals, McGraw Hill, 1988, ISBN - 0-07-028637-X

[14] Fernández A. V., Mastering OpenCV 4 with Python. *A practical guide covering topics from image processing, augmented reality to deep learning with OpenCV 4 and Python 3.7*. 2019. ISBN 978-1-78934-491-2.

[15] Rokem A., Machine learning for OpenCV. Packt. 2017. ISBN 978-1-78398-028-4

[16] Joshi P., Escrivá D. M., Godoy V., OpenCV by Example. Packt Publishing. 2016 ISBN 978-1-78528-094-8.

[17] https://numpy.org, NumPy package, accessed on 16/03/2020.

[18] E. Jannik Bjerrum. SMILES Enumeration as Data Augmentation for Neural Network Modeling of Molecules. *ArXiv e-prints*, Mar. 2017.

[19] Kearns, M., 1996. "A bound on the error of cross validation using the approximation and estimation rates, with consequences for the training-test split," *Advances in Neural Information Processing Systems,* vol. 8, pp. 183–189, Cambridge, MA: MIT Press.

[20]https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#, accessed on 20/03/2020.